VZ LINK magazine

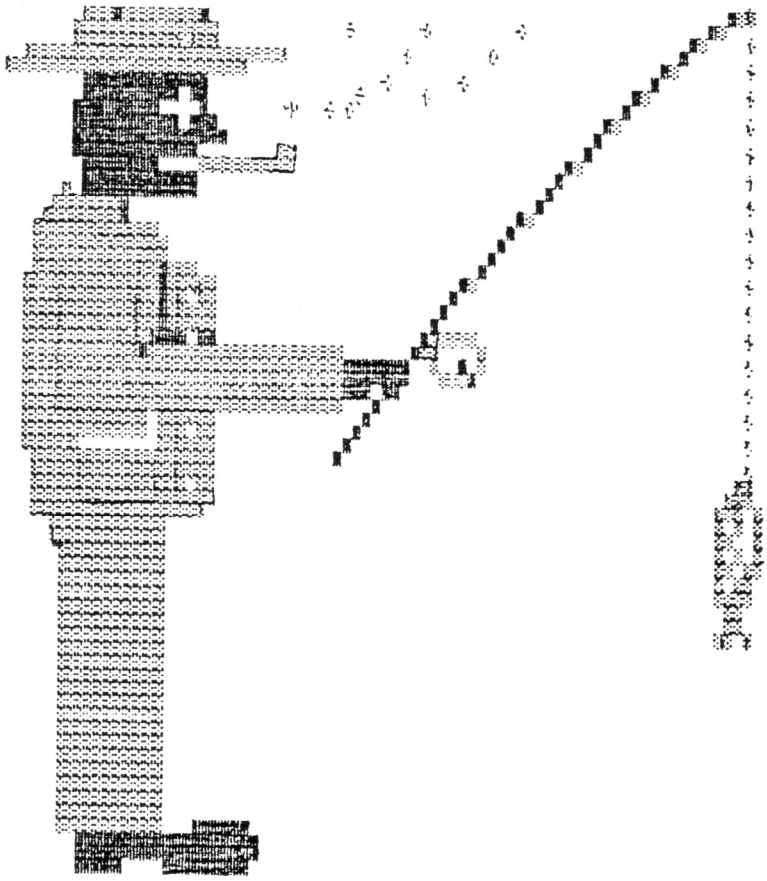
New Zealand VZ 300/200 Computer users club

ISSUE 44 JUNE 1990

PEDRO

THE

FISHERMAN



FROM ME TO YOU

This issue we have a superb program called *FISHERMAN* written by Ken Clarke.
We also continue our **Micro Magic** workshop. This month we have a brief look at the Printer Plotter . Unfortunately there are not many printer plotters around. I will bring mine to the next workshop if enough people are interested.

We also have an item on error handling. We hope to explore this subject further, sometime in the future. The Battery back-up Ram item also open's up some very interesting possibility's.

Lastly we have another Maillist update written by Peter Patterson. These updates, just go to show that all programs (well, nearly all) can be changed, updated or completely re-written to suit your needs.

NEXT ISSUE

We bring you Donkey Kong, a game for all the game busters out there. I will be interested to see the high scores that come in for this one.
In Micro magic we have a look at High Resolution programming.

I am not sure what else we will sqeeze in, but I am sure you will enjoy it.

HIGH SCORE COMPETITION

This competition is open to all VZ gamers. All you have to do is send in your high scores.
At the workshop meetings we will have a play off between the two top scores for each game entered on our list.
There will be a suitable prize for the winner.Any game can be entered on the list, so go to it.

BUY SELL SWAP

WANTED TO BUY OR SWAP
VZ printer interface, will buy or swap for Ram pack.
Contact Peter Hill ph 761-451

I am now using the Harrison wordprocessor patch supplied by Bob Kitch. This patch allows you to change printer codes in the middle of a line. I have used it on this page so you can see what it's like. You can purchase this patch from Bob Kitch for A$10.00 plus p & p.

P.O BOX 1972 C.P.O AUCKLAND N.Z

(C) AUCKLAND VZ 300/200 COMPUTER USERS CLUB (1986)
NEW ZEALAND VZ 300/200 COMPUTER USERS CLUB (1989)

ALL MATERIAL IN THIS NEWSLETTER IS SUBJECT TO COPYRIGHT.
CONTRIBUTED PROGRAMS & ARTICLES CAN BE REPRODUCED ON THE UNDERSTANDING THAT IT IS FOR THE PRIVATE USE OF SUBSCRIBERS ONLY.
COPYRIGHT IS ALWAY'S RETAINED BY THE AUTHOR.
THIS CLUB WILL NOT BE HELD RESPONSIBLE FOR ANY BREECHING OF COPYRIGHT CAUSED BY THE PUBLICATION OR USE OF ANY MATERIAL SENT IN UNDER A FALSE NAME.

YOU ARE WELCOME TO PHONE ME IF YOU HAVE PROBLEMS.
PHONE 761-451 AFTER 6.00 PM.

><(0) The FISHERMAN (0)><

Planning on a bit of fishing? Now is the time to get in a little practice.

Test your skills and cast a good line.

But be sure you have carefully calculated both the depth and the distance.

Full screen, animated graphics and an up-date on how your catch is comming on is displayed.

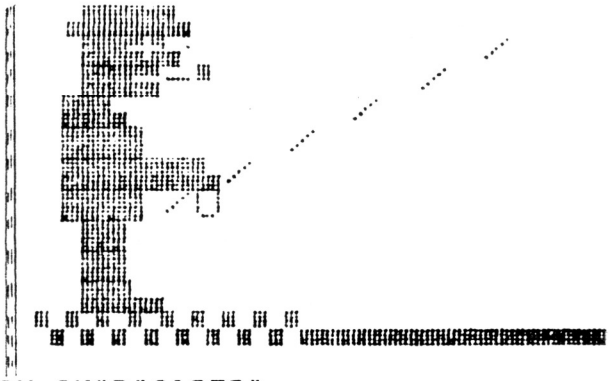
When a fish is caught, it thrashes about on the surface as it is hauled ashore and ends up in the fisherman's basket.

```
5 ' "FISHERMAN"
10 '"COPYRIGHT 1983 KEN CLARKE"
20 CLS:COLOR,1:GOSUB500:COLOR1,0
30 S=0:D=0:WEI=0
40 FORG=0TO100:NEXTG:CLS
50 PRINT@0," "    "      ; [F J J D]
60 PRINT"          /
70 PRINT"        /
80 PRINT"        /
90 PRINT"      /
110 PRINT"  "/
120 PRINT"  "/O
130 PRINT"
140 PRINT"
160 PRINT"        "
170 PRINT"                                                    ";
180 PRINT"                                                    ";
190 PRINT"                                                    ";
200 PRINT"                                                    ";
210 PRINT"TOTAL                                               ";
220 PRINT"WEIGHT="WEI"LBS";:PRINT@494,"                        ";
230 Q=RND(18)+9:O=RND(6)+10
235 IFO=16ANDQ=27,Q=26
240 PRINT@O*32+Q-32,"";
250 PRINT@83,"DEPTH";:INPUTB
260 PRINT@83,"DISTANCE";:INPUTA
270 IFB+11=OANDA<Q+1ANDA>Q-1,S=S+1
280 D=D+1
290 X=15:FORZ=2TOB+11:PRINT@Z*32+X-31,".";
300 X=X+((A-15)/(B+11)):NEXTZ
310 IFINKEY$="",320
320 X=15:FORZ=2TOB+11:PRINT@Z*32+X-31," ";
330 X=X+((A-15)/(B+11)):NEXTZ
340 PRINT@83,"              ":PRINT@11,S" FISH IN"D"CASTS"
350 IFB+11=OANDA<Q+1ANDA>Q-1,400
360 PRINT@168,"BAD LUCK THAT WEIGHED    "
370 PRINT@203,RND(5)"LBS"RND(14)"OZS   "
380 FORW=0TO800:NEXTW
390 GOTO50
400 FORT=QTO10STEP-1
410 PRINT@O*32+Q-32,"";:PRINT@320+T,"";:NEXTT
420 PRINT@330,""
430 PRINT@293,"(O>><":I=RND(14):U=RND(3)+1
440 PRINT@168," WEIGHT = "U"LBS"I"OZS   "
450 FORW=0TO250:NEXTW
460 WEI=WEI+U+(I>7)
470 GOTO50
```

The FISHERMAN.

by Ken Clarke.

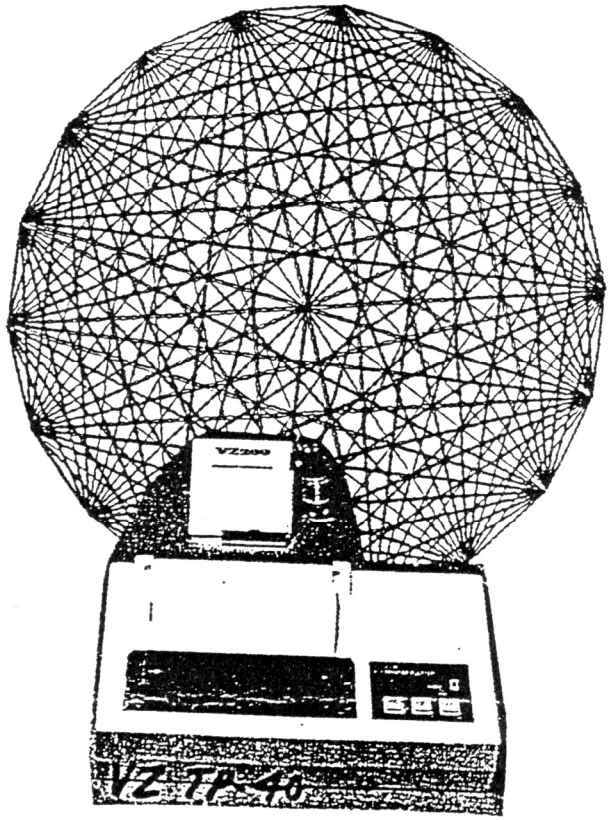
; Note — My printer is printing many of the graphics inside out (see code beside line 50)



```
500 DATA26,4,23,2,21,2,19,2,21,2,23,2,19,2,14,4,14,4,14,2,21,5
510 DATA14,4,14,4,14,2,26,5,23,7
520 DATA26,4,23,2,21,2,19,2,21,2,23,2,19,2,14,4,14,4,14,2,21,5
530 DATA14,4,14,4,14,2,26,5,19,7
540 DATA28,3,26,1,24,2,26,2,28,2,31,2,31,4,24,2,28,2,28,4,19,2
550 DATA24,2,24,4,23,3,21,1,19,2,21,2,23,2,21,2,19,2,23,2,26,7
560 DATA26,4,23,2,21,2,19,2,21,2,23,2,19,2,14,4,14,4,14,2,21,5
570 DATA14,4,14,4,14,2,26,5,23,7
580 DATA28,3,26,1,24,2,26,2,28,2,31,2,31,4,26,3,24,1,23,2,24,2
590 DATA26,2,31,2,31,4,24,3,23,1,21,2,23,2,24,2,26,2,28,2
595 DATA30,2,31,9
600 A$="[redacted]                    "
610 B$="BY                                  )<(O)"        ;Note - spacing
620 C$="  KEN    CLARKE                 "                  is critical
630 D$="     PRESS <SPACE> TO START"
640 POKE30777,1
650 FORI=1TOLEN(A$):PRINT@38,RIGHT$(A$,I):READS,T:SOUNDS,T:NEXTI
660 FORI=1TOLEN(B$):PRINT@110,RIGHT$(B$,I):READS,T:SOUNDS,T:NEXT
670 FORI=1TOLEN(C$):PRINT@200,RIGHT$(C$,I):READS,T:SOUNDS,T:NEXT
680 PRINT@354,"INPUT DEPTH OF CAST [0 TO 5]"
690 PRINT@384,"INPUT DISTANCE OF CAST [0 TO 30]";
700 PRINTD$:POKE30777,35
710 PRINT@261,"   [redacted]    (O)><":PRINT@261,"   )<(O)    [redacted]"
715 FORI=1TO10:NEXT
720 IFINKEY$<>" ",710ELSERETURN
```

Studio Adlib

MICRO MAGIC WORKSHOP



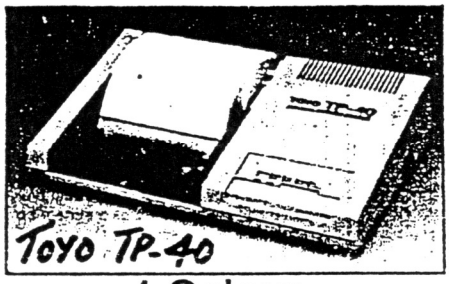
GOTTA PLOTTER ???

Those lucky enough to own a VeeZed or Toyo 4 - colour Printer/Plotter will find it a most versatile and reliable addition to their computer. The Toyo appears to be similar to the VeeZed PP 40 plotter except that the DIP switches are more accessible. A plotter is most suitable for use with the MICRODATA program. Lists of VeeZed programs held on cassettes can be printed using the 80 - character per line setting. Although the print is small it is quite leqible. Also, used with MICROTYPE it will produce both Upper and Lower case letters and act just like a big printer. Here are a few workshop notes to assist in operating these two plotters...

SOME PLOTTER PROGRAMS

If you want to make your Plotter draw the design which appears at the top of this Studio Ad Lib, then use the following program. You select the number of points in the design.....

```
350 CLS:LPRINTCHR$(18):LPRINT"C4":PI=3.1416
352 LPRINT"M220,-300";:LPRINT"I"
354 INPUT"NUMBER OF STAR POINTS";N
356 DIMA(N),B(N):FORT=0TO(2*PI-.001)STEP(2*PI/N):Z=Z+1
358 A(Z)=COS(T)*200+22
360 B(Z)=SIN(T)*200+22
362 NEXT
364 FORS=1TON-1:LPRINT"M";A(S);",";B(S)
366 FORD=S+1TON-1
368 LPRINT"D";A(S);",";B(S)
370 LPRINT"D";A(D);",";B(D)
380 NEXT:NEXT:LPRINT"A"
```



4 Colour
Printer/Plotter

GLOBE

```
10 ' ### GLOBE ###
20 LPRINTCHR$(18)
30 G=0:R=240:PI=3.14
40 '
50 LPRINT"C1"
60 FORJJ=0TO500:NEXTJJ
70 FORP=PI/31TOPI STEP PI/31
80 G=0
90 FORTH=0TOPI*2 STEP PI/16
100 X=R*SIN(TH)*COS(P)
110 Y=R*COS(TH)
120 Z=R*SIN(TH)*SIN(P)
130 GOSUB 340
140 IF Z3<0 THEN G=0:GOTO 160
150 GOSUB 410
160 NEXT TH
170 NEXT P
180 '
190 LPRINT"C2"
200 FORJJ=0TO500:NEXTJJ
210 FORQ=0TOPI STEP PI/31
220 G=0
230 FOR TH=0 TO PI*2 STEP PI/16
240 X=R*SIN(TH)*SIN(Q)
250 Y=R*COS(Q)
260 Z=R*COS(TH)*SIN(Q)
270 GOSUB 340
280 IF Z3<0 THEN G=0:GOTO 300
290 GOSUB 410
300 NEXT TH
310 NEXT Q
320 LPRINT"A"
330 END
340 '
350 Y3=Y*.877583-Z*.479425
360 Z1=Y*.479425+Z*.877583
370 X3=X*.825336-Z1*.564642
380 Z3=X*.564642+Z1*.825336
390 RETURN
400 '
410 GX=240+X3:GY=-240+Y3
420 IF G=1 THEN 440
430 LPRINT"M"+STR$(GX)+","+STR$(GY)
440 LPRINT"D"+STR$(GX)+","+STR$(GY)
450 G=1
460 RETURN
```

DEFINT, DEFSNG, DEFDBL and DEFSTR.

In your VeeZed is a VARIABLE DECLARATION TABLE. It is located between 30977 and 31002.
This VDT is 26 bytes - one for every letter of the alphabet. It is not accessible by VeeZed commands but it can be used. Here's how...

On switch-on the whole of the VDT is initialised to SINGLE PRECISION Variables. That means there is a 4 in every location. It is possible to change the 4s to other numbers to create other kinds of variables.
2 = INTEGER 3 = STRING 4 = SINGLE PRECISION 5 = DOUBLE PRECISION

INTEGER VARIABLES accept WHOLE NUMBERS. Example: A%
STRING VARIABLES accept CHARACTERS. Example: A$
SINGLE PRECISION VARIABLES accept numbers up to 6 decimal places. Example: A
DOUBLE PRECISION VARIABLES accept numbers to 14 or 16 decimal places. Example: A#

To change the VeeZed's VDT to make every letter, A to Z, into INTEGER VARIABLES, equivalent to DEFINT A-Z, the following routine will do it.

```
10 FOR I=30977TO31002:POKEI,2:NEXT
```

and the following can be used to define an individual variable...

```
10 POKE30912+ASC("A"),3
```

...this defines A as a STRING VARIABLE and no longer requires the $ sign.
Once a letter has been defined $, % and # are not needed.
The # was not accepted by the VeeZed anyway, but DOUBLE PRECISION is available using this method of changing the VDT. You can now devise a program to record your annual income and expenses without having the cents rounded off and producing the wrong totals which the VeeZed did using SINGLE PRECISION variables.

.....

ALPHABETICAL ORDER SORTING.

```
500 CLS:DIMA$(100):PRINT@37,"[illegible]"
510 PRINT:INPUT"  HOW MANY WORDS";A:CLS
520 FORI=1TOA:PRINT"WORD"I;:INPUTA$(I):NEXT
522 X=0
530 X=X+1
540 IFX>A,580
550 IFA$(X)="ZZZ",530
560 FORY=1TOA
562 IFA$(Y)<A$(X)THENX=Y
564 NEXT
570 PRINTA$(X):SOUND23,2:A$(X)="ZZZ":GOTO522
```

REVERSED REMS.

PUT YOUR REMARKS IN INVERSE BY PRECEDING THEM WITH QUOTATION MARKS:

```
10 REM"[illegible]
10 '"[illegible]
```

SOUND EFFECTS

ROM and RAM.

ROM is that part of a computer's memory that has been set by the makers, and is referred to as READ ONLY MEMORY
RAM being RANDOM ACCESS MEMORY means you can change the information held here.
Suppose we PEEKed ROM and POKEd whatever we found into RAM, then if we changed the information and used RAM instead of ROM, it would have the same effect as if we had altered ROM. And this is exactly what we have done to change the SOUND routine that was set into ROM by the makers. Then by changing the copy of the routine in RAM we can begin to invent SOUNDs that were not possible using the fixed routine.

```
0 'SOUND EFFECT
5 DATA100,110,120,140,150
6 DATA160,170,180,190,200
7 DATA210,220,240,250,255
10 FORI=0TO13:A=PEEK(I+13392):POKE -32768+I,A:NEXT
20 POKE30862,0:POKE30863,128
30 POKE-32763,20
40 FORJ=1TO15:READP:POKE-32766,P:X =USR(X):NEXT
50 RESTORE
60 GOTO40
```

By changing the DATA using numbers between 0 and 255 and the J LOOP to FORJ=1TO10... to suit the number of DATA items, you may invent some weird and wonderful SOUND EFFECTS like this one....

```
0 'RAY GUN
5 DATA2,4,6,8,10,12,14,16,18,10, 22
6 DATA 24,26,28,30,32,34,38,40
10 FORI=0TO13:A=PEEK(I+13392): POKE-32768+I,A:NEXT
20  POKE30862,0:POKE30863,128:POKE -32768,20
30  FORJ=1TO15:READP:POKE-32766,P :X=USR(X):NEXT
40 RESTORE
50 GOTO30
```

NOTE: 0 in DATA is silent. The duration of sound can be altered by POKING -32763 with numbers between 0 and 255. Too short a duration may not be satisfactory. Can you invent the sound of an explosion?.
Give it a go. At least you've broken out of the limits of the SOUND command.

ANY NEW SOUNDS WILL BE PRINTED IN THE NEWSLETTER, IF YOU SEND THEM IN. (Peter)

SELF — DESTRUCT

```
0 REM
10 CLS:PRINT@75,"MICRO MAGIC"
20 FOR I=1 TO 1000:NEXT
30 POKE31469,187
40 RUN
```

LEAP YEAR OR NON-LEAP YEAR

```
200 FORI=1TO5
210 CLS:LY$="LEAP YEAR":PRINT@70,"ENTER YEAR";:INPUTY
220 IFY/4<>INT(Y/4)THENLY$="NON-LEAP YEAR":GOTO240
230 IFY/100=INT(Y/100)ANDY<>INT(Y/400)THENLY$="NON-LEAP YEAR"
240 PRINT:PRINTTAB(6)LY$:FORT=1TO1500:NEXT
250 NEXT:FORT=1TO1500:NEXT:SOUND30,2:RUN
```

ERROR HANDLING

ON ERROR GOTO

As a result of my comment about the RESUME command last month two readers have replied with methods for getting into an error handling routine.
From C. Hudson, if A is the first line number of the error handling routine then

```
ON ERROR GOTO A   is implemented by the following

POKE 30961,INT(A/256) : POKE 30960,A-256*INT(A/256)
```

However 30960 and 30961 are set to zero when RUN is used so the POKE's should be done from within the BASIC program.

From S. Debreceny two practical examples of his method for entering an error handling routine. The following is quoted from his letter.

Now, I would like to show you how I have been using error traps in my programming, and have included some listings of extracts from programs that you can enter and see working.

Error trapping of some errors can be very helpful in exchange of data between user and computer and also can speed up your MODE(0) and MODE(1) graphics programs enormously. The two listings I have enclosed are examples of both of these facilities. Firstly, I'll deal with user/computer data exchange with an example of the division by zero error trap.

Whilst running a program, a user may encounter a request for numeric data (ie INPUT A). If a mathematical function needs to be applied to variable A, it may be found that a value of A has been entered that could cause a divide by zero subsequently a DIVIDE BY ZERO error, which halts the program, will be displayed leaving the user bewildered as to what he has said to upset the microprocessor.

So, instead, try trapping the error, go back to the input and give the mug, sorry, user a second chance. To do this we need to look at our 'invisible commands'. We can see from the table (in issue,41 ED) that the token for ERROR is 158.
The VZ 200 token 158 is SOUND.

You must take my word for this, that indeed when token 158 stands alone it is interpreted as SOUND. However if token 158 is preceeded by token 161, which is ON it is interpreted as ERROR.

```
e.g. line 10  158 20,2        is SOUND 20,2
     line 10 161 158 141 100 is ON ERROR GOTO 100
```

The divide by zero error trap example listing I have included picks a random number between 0 and 10. It then divides 100 by this number and prints the function on the screen. Ultimately 0 will be selected as a random number, and as the computer attempts to divide 100 by 0, generates a DIVIDE BY ZERO error. This error is trapped, the error type and line in which the error occurred is printed and the program resumed at the line which selects a new random number.

Now for the ILLEGAL FUNCTION trap. This may take a little more explanation as to why it is employed, but I'm sure you will see how it will speed up your graphic programs considerably. The program listing is an extract from a game I have written. It is a MODE(1) graphics game and the section we are looking at involves an object that moves across the screen shooting bullets randomly in all directions.
First lets look at how we could do this without the error trap.

The program would randomly select a direction, it would then SET a point in that direction, check if off screen, SET the next point and RESET the first, check if off screen etc etc. The off screen check every time a point is to be SET is unnecessary and wastes lots of valuable microsecond in the attempt to approach real time graphics simulation.

By using the ILLEGAL FUNCTION error trap, the bullet is permitted to advance as it will until the stage is reached when SET command can not be executed because of an off screen co-ordinate. An ILLEGAL FUNCTION error occures, the error is trapped and the co-ordinates are reset for a new bullet. This allows those microseconds to be used for collision checking with other objects.

BASIC PROGRAM TO DEMONSTRATE ERROR TRAP OF
DIVIDE BY ZERO ERROR

```
10 CLS
12 ON ERROR GOTO100
14 LET A=INT(RND(0)*10)
16 PRINT"RANDOM NUMBER = 100/";A;"=";INT(100/A)
18 SOUND20,1;0,8:GOTO14
100 PRINT:PRINT:PRINT"*** ERROR TRAP ***"
102 PRINT" ERROR TYPE ";ERR/2+1
104 PRINT" IN LINE NO. ";ERL
106 PRINT" A=";A:SOUND30,1;0,8
108 PRINT:RESUME14
```

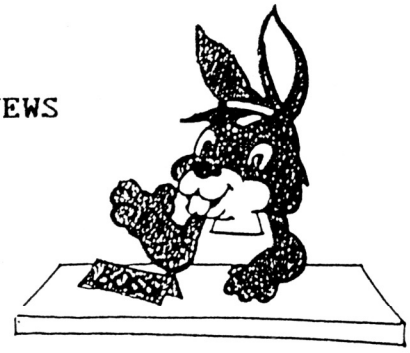
BASIC PROGRAM TO DEMONSTRATE ERROR TRAP OF
ILLEGAL FUNCTION ERROR

```
10 MODE(1):COLOR2
12 ON ERROR GOTO100
14 SX=4:SY=60:SET(SX,SY)
16 X=SX:Y=SY
18 A=RND(0)*2-1:IFABS(A)<.05THEN18
20 B=RND(0)*-1.8:IFABS(B)<.5THEN20
22 X=X+A:Y=Y+B
24 SET(X+A,Y+B)
26 RESET(X,Y)
28 SX=SX+.1:SET(SX,SY):RESET(SX-1,SY):GOTO22
100 IFSX>127THENRESET(SX-1,SY):RESET(X+A,Y+B):RESUME14
102 RESET(X,Y):RESUME16
```

BASIC PROGRAM DISASSEMBLY UTILITY
ADD TO BOTTOM OF PROGRAM AND THEN RUN 1000

```
1000 NL=31465
1002 LPRINT
1004 LPRINT"LINE NO. ";PEEK(NL+2)-256*PEEK(NL+3);TAB(16);
1006 FORI=NL+4TONL+255
1008 IFPEEK(I)=0THENLPRINT:GOTO1016
1010 IFPEEK(I)<32ORPEEK(I)>90THENLPRINTPEEK(I);:GOTO1014
1012 LPRINTCHR$(PEEK(I));
1014 NEXT
1016 NL=PEEK(NL)+PEEK(NL+1)*256
1018 K$=INKEY$:K$=INKEY$:IFK$=""THEN1018ELSESOUND20,1:GOTO1002
```

REPRINTED FROM VE ZEE NEWS
AUGUST 1984



Memories are made of this

By Gordon Findlay

Memory is one of the most important components of a computer system. No memory — no computer. Let's look at computer memory, and try to fight our way through the jargon.

At a really fundamental level, there are two sorts of memory: main and backing stores.

Main store — sometimes called core store, main memory or whatever — is the memory the computer system can get at or access directly. It is built into the computer's main unit.

Backing storage is not directly accessible. It consists mainly of tapes and disks. Any information or instructions in this sort of memory must be read into main memory before it can be used.

Why "core" storage? The earliest computers, from the mid-1940s on experimented with many types of storage. Wiliams tapes (just like a TV tube) and containers of mercury with circulating electrical pulses were two of the chief contenders, until stores formed by stringing together little rings ("cores") of magnetic material were built. These proved most successful, and this was the origin of the term. All core store is now made from semi-conductor chips, but the name lives on.

From here on, we will be concerned only with main memory. In small computers, this also comes in two main types — the infamous ROM and RAM. No two "words" strike more fear into the computer tyro.

Permanent memory

Some memory is permanent and never loses its contents. This is ROM (Read Only Memory). Even when the computer is turned off the contents of ROM are preserved. This is ideally suited to the most important instructions and data the computer needs always — the operating system and languages (see the article on operating systems in the August *Bits & Bytes*. Without ROM, we would have to load BASIC into our machines every time we turned them on. From tape that might take four to 20 minutes. One bug, and we might have to reload! So ROM is typically used to store the BASIC language, operating system and such important features. The name? This type of memory can be read only, not written into.

ROM is "programmed" or filled in its manufacturing process, unlike most other forms of memory which are filled by (or with) a program as it runs.

The other main sort of memory is called RAM (Random Access Memory). This really is a stupid name — ROM is "random access" too. Which just means that any particular location in memory can be read independently without reaching all those before it. A move was made to call RAM "Read/Write" memory, but it seems to have petered out.

RAM can be written too. This means the program can store things in RAM, as well as reading them from it. This sort of memory is used to store a program and the data it is working on. If you are using a program to store and look up telephone numbers, you will have stored in a few RAM locations names (say *Bits and Bytes)* and in other locations the corresponding phone numbers ((03) 66566) and so on. Other locations will keep the program instructions themselves.

Machines differ in the amounts of ROM and RAM they include. Roughly speaking, the bigger machines use little or no, ROM; the smaller ones more.

Static & dynamic

There are several other types of memory you will hear about. RAM can be either static or dynamic. This refers to the way the chips behave electrically, and makes no difference to the way they are used. PROM is programmable ROM — ROM which can be loaded with data or information after it has been made. This involves melting ("fusing" or "blowing") small links in a chip, and is done with a piece of hardware called a PROM "blower". EPROM is Erasable PROM — programmable ROM which can be wiped clean and reused. The erasing is done by ultraviolet light, unless the PROM is also EEPROM (electrically erasable PROM). Erasing is then done with a similar, or the same, piece of hardware.

Uses of PROMS? Sometimes it is desirable to change some of the really low level information in a system. I have had the character set of my computer (the actual shades of the letters which appear on the screen) changed. This was done by loading the shapes I wanted, suitably coded, into an EPROM, and replacing one of the ROM chips which came with the computer originally.

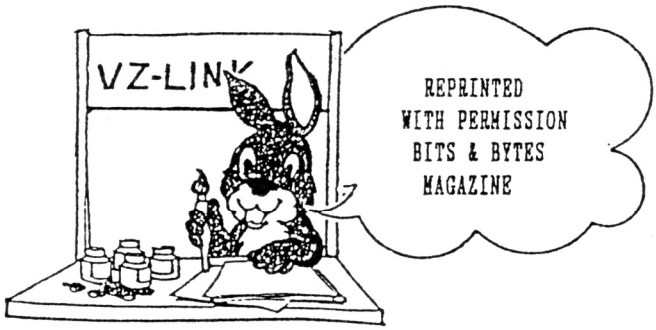
Memory is divided into bytes — more or less the amount of memory which the computer uses at one time. The number of bytes is limited by the design of the processor and the rest of the system. The amount of memory is measured in "K". A larger personal computer will contain 64K of memory, some ROM and some RAM. 1K is 1024 bytes, so 64K is 65536 bytes which, just to add to the confusion, will be numbered 0 to 65535. The abbreviation, "K" is sometimes taken to mean 1000, rather than 1024, so the 64K might be written as 65K. The moral of this confusing mish-mash is that the exact numbers are not important.

Computers specify their memory sizes in different ways. For example, a 48K TRS-80 machine has 64K of memory in all. The 48K refers to the amount of user memory — memory (RAM, in fact) available for use by programs. This doesn't include memory used by the operating system, or ROM. On the other hand, the Commodore 64 has the same amount of memory in total, but only roughly 38K is available for programs. The total amount of memory in a system will be divided into four parts — ROM, System RAM, video RAM (used to contain the screen display) and user RAM. Make sure you are comparing like figures — the figure for user RAM is the important one.

How much memory is enough? The early machines had very small main memories. In 1950, 512 bytes (0.5k) was regarded as a lot. Computers are available with as little as 1K, and as much as 512K. The amount you need is determined by the sort of programs which you want to run. To run the CP/M operating system, you'll need 64K. The famous program, Lotus 1-2-3, requires 192K from memory. On the other hand, the ZX-81 has shown that games and simple programming at least are possible in just 1K.

Another common phrase is "virtual memory" In this set-up, a machine pretends to have a very large or infinite memory by using disk as if it were a large block of RAM. This is done by the operating system, and is most common in large machines.

All this memory and I still can't remember my kid's birthdays!



BATTERY BACK UP

BATTERY BACK-UP RAM BY DAVE BOYCE

This description will cover Battery Back-Up static RAM in any location, although this article is set up for (decoding only) in the 6000H to 67FFH area. That is the 2K gap left by the Disk Controller. Non Disk users can put an extra 8K RAM in the 4000H to 5FFFH area. Inside the VZ200 the 74LS139 has two unused decoded outputs, Pin 6 (8K Block) and Pin 12 (2K Block). Pin 12 is the one used in this article although an alternate decoder (74LS138) is shown for persons not wishing to open up their VZ200, or for VZ300 owners.

The operation of DATA retention is fairly easily achieved. Refering to the circuit diagram locate the 4 Way DIP Switch. You will find that only 3 Switches are used for the Battery Back-Up feature. S3 (Switch 3) is not used and no connection is made to it.

S1 - Controls the main power supply from the VZ to the RAM.
S2 - Controls the +3V (2 X AAA) Battery Back-Up Supply.
S3 - No connection to this Switch and is not used.
S4 - ENABLES or DISABLES the WRITE line from the VZ to the 2K RAM Chip.

Ensure that S1 is ON when the VZ is ON or You will not be able to WRITE to the RAM.

To WRITE to the RAM S1 and S4 must be ON.

To PROTECT RAM DATA while the VZ is ON, turn OFF S4.

To PROTECT RAM DATA while the VZ is OFF :- Before turning the VZ OFF ENSURE that S2 is ON and S4 are OFF.

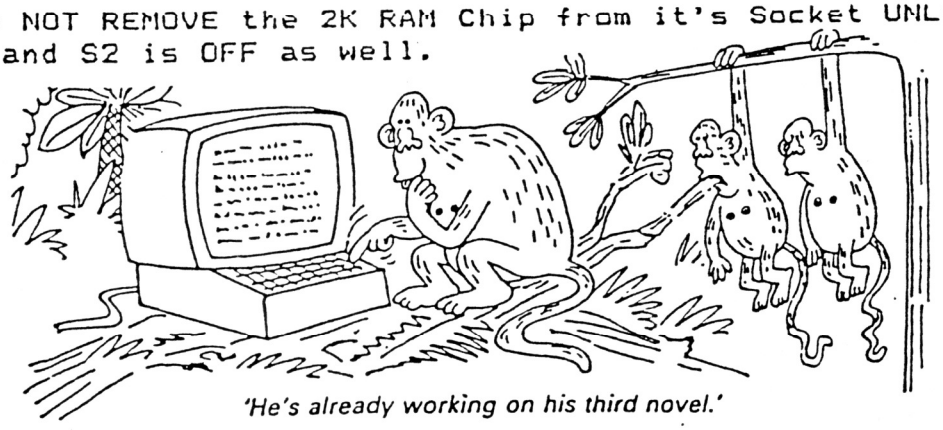
To ERASE DATA while the VZ is OFF, turn S2 OFF.

To ERASE DATA while the VZ is ON :- Turn S1 and S2 OFF. Wait approx. 10 sec's. before turning S1 back ON. S2 is optional.

NOTE -- Before REMOVING or REPLACING the Batteries ensure that S2 is in the OFF position.

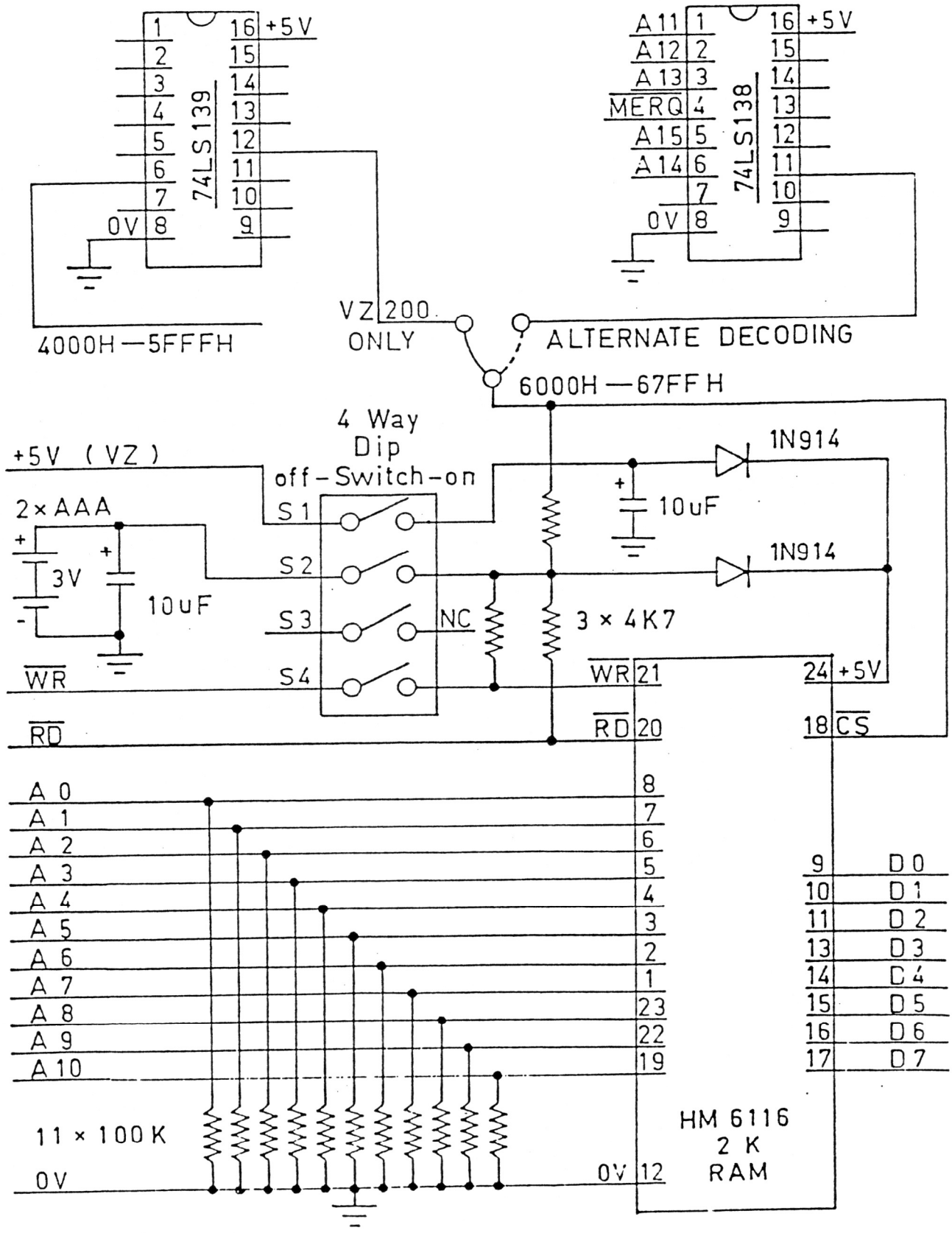
Use ALKALINE batteries only. (NOT Nickle Cadmium).

NOTE -- DO NOT REMOVE the 2K RAM Chip from it's Socket UNLESS the VZ is OFF and S2 is OFF as well.



'He's already working on his third novel.'

VZ 200 / 300 BATTERY BACK UP RAM



© COPYRIGHT DAVE BOYCE

TO WRITE WITH VZ ON			
S1	ON	(+5V)	
S2	X	(+3V)	
S3	X	(NC)	
S4	ON	($\overline{WR}$)	

TO PROTECT WITH VZ ON OR BEFORE TURNING VZ OFF			
S1	ON	(+5V)	
S2	ON	(+3V)	
S3	X	(NC)	
S4	OFF	($\overline{WR}$)	

TO ERASE WITH VZ ON			
S1	OFF	(+5V)	
S2	OFF	(+3V)	
S3	X	(NC)	
S4	X	($\overline{WR}$)	

TO ERASE WITH VZ OFF			
S1	X	(+5V)	
S2	OFF	(+3V)	
S3	X	(NC)	
S4	X	($\overline{WR}$)	

X = DONT CARE $\overline{WR}$ = WRITE NC = NO CONNECTION

MAIL LIST UPDATE

by Peter Patterson

A simple update to the maillist program to enable the use of two disk drives.

I always backup files on another disk when using the excellent Joe Leon updated MAILLIST, but have found it a nuisance swapping discs around.

If you have two disc drives, the following simple update makes saving data on two discs easy.

```
850 GOSUB40200:PRINT@354,SP$:PRINT@380,"N"
885 GOSUB40200
5240 NEXT:GOSUB20:DRIVE1
6140 GOSUB20:DRIVE1:GOSUB7000:GOTO600
40200 PRINT@354,"[illegible]";:SOUND30,1
40205 D$=INKEY$:D$=INKEY$:IFD$=""THEN40205
40210 IFD$="1"THENRETURNELSEIFD$="2"THEN40215
40215 DRIVE2:RETURN
```

Lines 850 and 885 send the load or save data routines firstly to the **'DRIVE SELECT'** subroutine.
Additions to lines 5240 and 6140 ensures the computer is switched back to DRIVE 1 after data is saved or loaded.
Lines 40200 to 40215 are the Drive select routine"